プログラム言語を使った日本語テキスト処理技術について

An Technical Comment for Japanese Text Mining using Programming Language

廣島　勉

1 序

今回、著者は椿田・早川両氏による共同研究 ([1]) に、電子形式のテキストデータの処理に於ける技術面から協力した。両氏はそれぞれの専門分野から、テキストマイニングの技術に多大な関心を持たれていて、この方面でのプログラム言語の活用の方法についての助言を著者に求められた。著者は、その活用事例の詳細を述べることで、施した処理がデータの正当性を損なうものでないことを明確にする必要があると考えた。また、両氏との対話の中で、コンピュータ技術的に容易であることの認識において、著者と両氏の間にギャップがあることが感じられた。データ処理の実際を記録することが、テキストデータの書式についてコンピュータ技術面から考慮すべきことを示唆することとなり、テキストデータ収集方法等の検討の際のコンピュータが専門ではない研究者の助けになればと考える。

2 プログラム言語 Perl

今回、著者が用いたプログラム言語は Perl である。元々がテキスト処理の豊富な命令をもち、有志によるユーティリティが多数提供されていて、これらを活用すれば、一般的なテキスト処理の問題は数十行のプログラムコードで解決できることを経験上知っているからである。

また Perl は Web サーバーで CGI プログラムを記述する言語としても利用されている。従ってアンケート収集の Web ページを開発し、データの収集・解析に応用することも考えられる。

今回は、表計算ソフトウェアで作成したテキストファイルを Perl 言語を使って処理した。

本節で Perl 言語の基礎的な文法要素について解説する。実際は、データに含まれる特殊な文字への対応や、Perl 言語のプログラムの堅牢制を高めるために推奨されるスタイルでプログラムコードを記述したが、簡単のため、以降ではそのような部分は割愛した。

2.1 Perl の変数

プログラム言語を使ってデータを操作するためには、まず変数に当該データを格納し、その変数を経由してデータを操作するようプログラムコードを記述する。Perl 言語の変数には、スカラー変数、配列変数、ハッシュ変数の 3 種類があるが、本論文ではスカラー変数と配列変数を取り上げれば十分であろう。

2.1.1 スカラー変数

単一のスカラー値データを格納する変数である。スカラー値データは数値でもテキストでも良い。§ 2.2 でも述べるように、Perl モジュールのインスタンスを格納することもある。Perl 言語では変数を識別する変数名の前に $ をつけて表現する。

2.1.2 配列変数

順序のついた複数のスカラー値データを格納する変数である。番号を指定することで、個々のスカラー値データを参照、変更可能である。Perl 言語には配列変数の一部を取り出したり、他のデータに置き換えたり、データを追加、削除するための命令が豊富である。Perl 言語では変数を識別する変数名の前に @ をつけて表現する。

■例 入力

```
Sun:Mon:Tue:...
```

のようにコロンを含むテキストデータを 1 行分読み込み、コロンをコンマに置き換えて

```
Sun,Mon,Tue,...
```

と出力するプログラムコードの断片は次の様になる。split と join は、スカラー値データと複数のスカラー値データの間の変換をする命令である。

```
1  $text = <STDIN>;
2  @list = split(m/:/, $text);
3  $text = join(',', @list);
4  print $text;
```

1. スカラー変数 $text にテキストデータを読み込み、
2. コロンで複数のデータに区切って、配列変数 @list に格納し、
3. コンマで再び 1 行のテキストデータに連結してスカラー変数 $text のテキストデータを更新し、
4. そのデータを表示する。

2.2 Perl モジュール

Perl 言語には豊富な拡張機能が CPAN (Comprehensive Perl Archive Network) [*1] に登録されている。ほとんどの拡張機能は Perl モジュールとして提供されている。Perl モジュールを使用するプログラムコードでの手続きはオブジェクト指向プログラムの作法に従い次の様になる。

```
1  use FileHandle;
2  $io = new FileHandle;
3  $io->open("data.csv");
```

1. FileHandle モジュールをロードする。
2. FileHandle インスタンスを生成する。
3. インスタンスに対し open メソッドを呼び出し、data.csv ファイルを読み込み専用で開く。

モジュールシステムではこの例の様にモジュールのインスタンスを生成して変数に格納し、以降はその変数を介してモジュールの提供する機能をメソッドとして呼び出す。記法は次の様に

*1 http://search.cpan.org/

なる。

```
$instance->method;
```

生成するインスタンスは外部プログラムへの参照であったり、データベースへの接続であったり、ファイルへの入出力チャンネルであったりと様々である。

3 形態素解析エンジン MeCab

データにはアンケート被験者が自由な形式で書いた文章が含まれている。テキストデータマイニングではこの文章を単語に分け、統計的な処理を行う必要がある。今回は被験者が筆記した文章をあらためてコンピュータに入力するのであるが、複数の入力担当者間で分割方法を統一するのは実際的でなく、入力の際に単語に分解することは無理がある。そこで京都大学情報学研究科と日本電信電話株式会社コミュニケーション科学基礎研究所によって開発されている形態素解析エンジン MeCab [*2] を使用する[*3]。MeCab には Perl 言語で使用するためのモジュールが公開されている。

`MeCab` モジュールを使うプログラムコードでは、§ 2.2 でも述べたようにまずインスタンスを作成する。

```
use MeCab;
my $mecab = new MeCab::Tagger;
```

`MeCab::Tagger` モジュールは `MeCab` モジュールをロードしたときに自動的にロードされるモジュールである。そのインスタンスをスカラー変数 `$mecab` に格納する。その後 `MeCab` とのデータのやり取りは `$mecab` を介して行う。
例えばスカラー変数 `$text` に

> 森の中の木や草の香り。

のテキストデータが格納されている場合、

```
$analysis = $mecab->parse($text);
```

で、スカラー変数 `$analysis` に MeCab による解析結果が複数行のテキストデータとして次のように格納される。

```
森      名詞,固有名詞,人名,姓,*,*,森,モリ,モリ
の      助詞,連体化,*,*,*,*,の,ノ,ノ
中      名詞,非自立,副詞可能,*,*,*,中,ナカ,ナカ
の      助詞,連体化,*,*,*,*,の,ノ,ノ
木      名詞,一般,*,*,*,*,木,キ,キ
```

*2 http://mecab.sourceforge.jp/
*3 早川・椿田両氏は形態素解析エンジン ChaSen も使用したとのことである。

```
や      助詞,並立助詞,*,*,*,*,や,ヤ,ヤ
草      名詞,一般,*,*,*,*,草,クサ,クサ
の      助詞,連体化,*,*,*,*,の,ノ,ノ
香り    名詞,一般,*,*,*,*,香り,カオリ,カオリ
。      記号,句点,*,*,*,*,。,。,。
EOS
```

「森」が固有名詞として認識されているが今回の分析では問題はない。この結果を行単位に分割して配列変数 @results に格納する。

```
1  @results = split(m/\n/, $analysis);
2  pop @results;
```

1. スカラー変数 $result を改行で分割して配列変数 @results に格納する。
2. 解析結果の最後に付けられる「EOS」が、@results の最後の要素となるので除去する。

解析結果の各行、例えば

```
香り    名詞,一般,*,*,*,*,香り,カオリ,カオリ
```

をスカラー変数 $result に順次格納しつつ単語と品詞分類を取り出す。

```
1  ($word, $attr) = split(m/\t/, $result);
2  ($type, $class) = split(m/,/, $attr);
```

1. $result に格納されたテキストデータをタブ文字で分割し、単語部分をスカラー変数 $word に、形態素種別部分をスカラー変数 $attr に格納する。
2. さらに形態素種別部分をコンマで分割し、最初の 2 要素、すなわち品詞と第 1 段の品詞細分類をスカラー変数 $type、$class に格納する。

以上の結果 $word には「香り」が、$type には「名詞」が、$class には「一般」が格納される。以降の処理は分析の目的によって変わる。例えば、スカラー変数 $text に格納された文章の名詞数を数えるプログラムコードは次の様なものとなる。

```
@results = split(m/\n/, $mecab->parse($text));
pop @results;
$count = 0;
foreach $result (@results) {
  ($word, $attr) = split(m/\t/, $result);
  ($type, $class) = split(m/,/, $attr);
  ++ $count if $type eq '名詞';
}
```

4 Text::CSV_XS モジュール

CSV (Comma Sepparated Values) フォーマットは一般的な表計算ソフトウェアが読み書きできるテキストデータの書式である。

処理前データはアンケート被験者の手書き回答を、複数の入力担当者が手分けして打ち込む[*4]ため、ごく一般的な入力環境を提供するソフトウェアを使用する必要があり、処理後データはコンピュータが専門ではない早川・椿田両氏が分析に使用するため、やはり一般的な分析環境を提供する表計算ソフトウェアで読み込めるファイル形式にする必要があった。

したがって、表計算ソフトウェアで製作した表を CSV フォーマットのファイルとして保存した。処理後ファイルも CSV フォーマットである。

■ CSV フォーマットの例　各行とも後部を省略 (...) している。

```
"ID","sex","nationality","age","jibika","olfactory","konomi",...
7,2,1,21,0,2," お香のにおい。",...
300,1,1,21,0,2," 森の中の木や草の香り。",...
```

表の書式は、行がアンケート被験者を表し、列がアンケートの設問を表す。各セルには被験者の設問に対する回答がテキストとして入っている。第 1 行は例によって設問のタイトルとなっている。本論文ではデータベースの用語を用いることとするので、ここで用語の対応関係を明確にしておこう。

レコード　被験者ごとのデータ。表の第 2 行以降の各行にあたる。
フィールド　被験者の設問に対する回答。セルにあたる。
ラベル　フィールドの定義。表の第 1 行の設問のタイトルにあたる。

Perl 言語で CSV フォーマットのファイルを扱うためのモジュールが `Text::CSV_XS` モジュールである。

```
use Text::CSV_XS;
$csv = new Text::CSV_XS;
```

以降は CSV フォーマットのファイルの第 2 行目以降、すなわち各レコードからフィールドのデータを取り出す。`$record` には第 2 行目以降のテキストデータが順次格納されるとする。

```
1  $status = $csv->parse($record);
2  @fields = $csv->fields;
```

1. `$record` のテキストデータを走査する。`$status` には `$record` のテキストデータが正しい CSV フォーマットであるのかが格納されるので、必要があればその値を調べてエラー処理をする。

*4 入力担当者による差異が書式に生じていたが，条件分岐により吸収できる程度であった．データ処理にプログラム言語を使用する利点の 1 つである．

2. 配列変数 @fields に当該レコードのフィールドが格納される。

ここで被験者の文章が格納されたフィールドに対し§ 3 で述べた様な処理を施す。例えば、その結果としてスカラー変数 $count に単語数が格納されたとしよう。これを当該レコードのフィールドの最後に追加するならば、次のように記述する。

```
push @fields, $count;
```

結果を再び CSV フォーマットのテキストデータとして出力するプログラムコードは次の様になる。

```
1  $status = $csv->combine(@fields);
2  print $csv->string;
```

1. combine メソッドでフィールドを CSV フォーマットのテキストに結合する。
2. そのテキストデータは string メソッドで参照する。

最後に単純な例を示そう。CSV フォーマットのテキストファイル data.csv からレコードを読み込み、最初のフィールドのテキストデータの一般名詞数を数えてフィールドの最後に追加し、再び CSV フォーマットで出力するには次の様なプログラムコードとなる。

```
use FileHandle;
use Text::CSV_XS;
use Mecab;

$input = new FileHandle("data.csv");
$csv = new Text::CSV_XS;
$mecab = new Mecab::Tagger;

$input->getline;
while($csv->parse($input->getline)) {
    @fields = $csv->fields;
    @results = split(m/\n/, $mecab->parse($fields[0]);
    pop @results;
    $count = 0;
    foreach $result (@results) {
      ($word, $attr) = split(m/\t/, $result);
      ($type, $class) = split(m/,/, $attr);
     ++ $count if $type eq ' 名詞 ' and $class eq ' 一般 ';
    }
    $csv->combine(@fields, $count);
    print $csv->string, "\n";
}
```

実際の作業では、各設問にたいする各回答ごとに語彙数を計算した表、被験者が挙げた商品名のリスト、被験者の ID と自立語のクロス表等を CSV 形式で作成した。

5 結び

Perl 言語と Perl 言語によるデータ処理については、[2]、[3]、[4] を参照してもらいたい。本論文ではテキスト処理に強みをもつ言語 Perl を扱ったが、Ruby、Python、Java 等、その他の言語でも基本的なアルゴリズムは同一となる。また、これらの言語はいずれも主要なオペレーションシステムで使用できる。形態素解析エンジン MeCab のインターフェースもこれらの言語用のものが公開されている。

データ処理にプログラム言語を使用する際のアプローチと、データ処理アプリケーションを使用する場合のアプローチとは本質的に異なる。後者ではデータをアプリケーションに読み込ませ、対話的に（もしくはメニューやボタン等のグラフィカルユーザインターフェースを介して）段階的に処理を施して、その度に処理結果を確認する。適宜中間段階でデータを保存し、誤った操作や、短絡的な操作で、データを破壊してしまった場合は、破壊されていない段階のデータから処理をやり直す。記録として、元データ、複数の中間データ、最終データができあがる。

それに対し前者では、最初は元データを単純に読み込みそのまま出力するプログラムから開始し、段階的にデータ処理命令をプログラムに追加しては、元データをプログラムにかけて、中間結果を出力しては確認する。この出力はファイルである必要はない。データがテキストデータである限り画面への出力で十分である。望む形のデータになったところで、最終データとして保存する。記録として残るのは、元データ、最終データと、元データを最終データに変換するための処理命令の手順、すなわちプログラムである。

このように、一過的なデータ処理では最初から完成されたプログラムを記述するわけではない。プログラムとして処理の手順を残しておくことで、同種のデータが追加された場合や、類似のデータを処理する場合に手間が低減される。

参考文献

[1] 椿田　貴史・早川　貴 , テキストマイニングによる香りに関する連想の分析方法－その手続き・意義について，NUCB Journal of Economics and Information Science vol.51 No.2 (2007)

[2] Larry Wall, Tom Christiansen & Jon Orwant, Programming Perl, 3rd Edition, O’Reilly & Associates, 2000

[3] David Cross (宮川　達彦訳), Perl データマンジング , ピアソン・エデュケーション , 2003

[4] Tom Christiansen & Nathan Torkington (Shibuya Perl Monges 監訳), Perl クックブック , 第 2 版 , オライリー・ジャパン , 2004